British sign language recognition in the wild based on multi-class SVM

Quinn, Matthew; Olszewska, Joanna Isabelle

Published in:
Proceedings of the 2019 Federated Conference on Computer Science and Information Systems

DOI:
10.15439/2019F274

Published: 04/09/2019

Document Version
Publisher's PDF, also known as Version of record

Proceedings of the Federated Conference on Computer Science and Information Systems pp. 81–86
DOI: 10.15439/2019F274
ISSN 2300-5963 ACSIS, Vol. 18

British Sign Language Recognition In The Wild Based On Multi-Class SVM

M. Quinn and J.I. Olszewska

School of Computing and Engineering
University of the West of Scotland, United Kingdom
Email: joanna.olszewska@ieee.org

***Abstract*—Developing assistive, cost-effective, non-invasive technologies to aid communication of people with hearing impairments is of prime importance in our society, in order to widen accessibility and inclusiveness. For this purpose, we have developed an intelligent vision system embedded on a smartphone and deployed in the wild. In particular, it integrates both computer vision methods involving Histogram of Oriented Gradients (HOG) and machine learning techniques such as multi-class Support Vector Machine (SVM) to detect and recognize British Visual Language (BSL) signs automatically. Our system was successfully tested on a real-world dataset containing 13,066 samples and shown an accuracy of over 99% with an average processing time of 170ms, thus appropriate for real-time visual signing.**

I. Introduction

THERE are 11 million people with hearing loss across the United Kingdom (UK), i.e. 1 in 6 people, with around 900,000 of these persons having profound hearing loss [1]. Users of British Sign Language (BSL) number in the 150,000 range, with more than half of them using BSL as their first language [2]. By stark contrast, there are far less registered BSL interpreters, with 1540 being publicly available on the National Registers of Communication Professionals working with Deaf and Deafblind People (NRCPD) [3].

BSL consists of 26 hand-shapes; one being correlated to each letter of the alphabet, as illustrated in Fig. 1. Each letter is formed using two hands except for the letter *'C'*, using only one hand [4].

With the current expansion of Artificial Intelligence (AI) in daily applications [5], intelligent systems can play an important role for sign language recognition (SLR).

However, despite a number of technologies developed for the automated, visual recognition of gestures within the field of Human Computer Interaction (HCI) [6], [7], only very few studies have tackled with automated BSL translation [8].

In HCI, most of the gesture recognition systems usually require special hardware equipments, such as depth camera [9] or gloves [10], which are usually not available outside a laboratory and/or have limited utility in the wild.

On the other hand, SLR systems integrating machine learning techniques such as genetic aglorithms (GA) [11] or convolutional neural networks (CNN) [12] have been mainly focused on the American Sign Language (ASL) [13], which uses static single-hand poses (as opposed to BSL which uses two-handed ones) to spell individual letters. Most of the available datasets are also only dedicated to ASL. It is worth noting that ASL and BSL languages have little crossover in terms of their actual constituent phonemes and are mutually incomprehensible to one another.

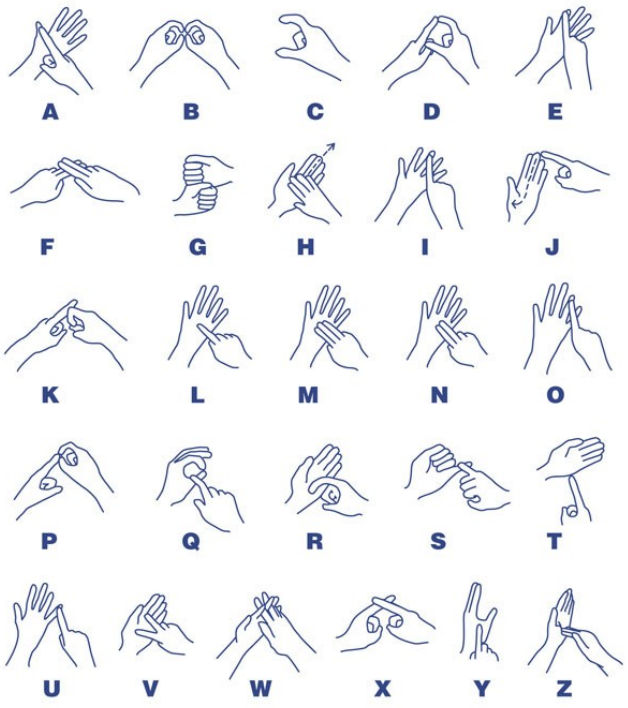


Fig. 1. Schematic overview of the British Visual Language (BSL) alphabet right-handed fingerspelling [2].

Hence, compared to ASL, BSL has a smaller body of research dedicated to visual recognition and as such, little online BSL datasets exist. Thus, methods based on deep learning [12], which requires hundreds of thousands of training data samples, are not directly available for BSL.

In this work, we propose the development of an accessible, intelligent vision system for real-time, automated BSL recognition in the wild. This assistive technology is inbuilt as a smartphone application, using computer-vision algorithms to process the images captured in real-time by the smartphone

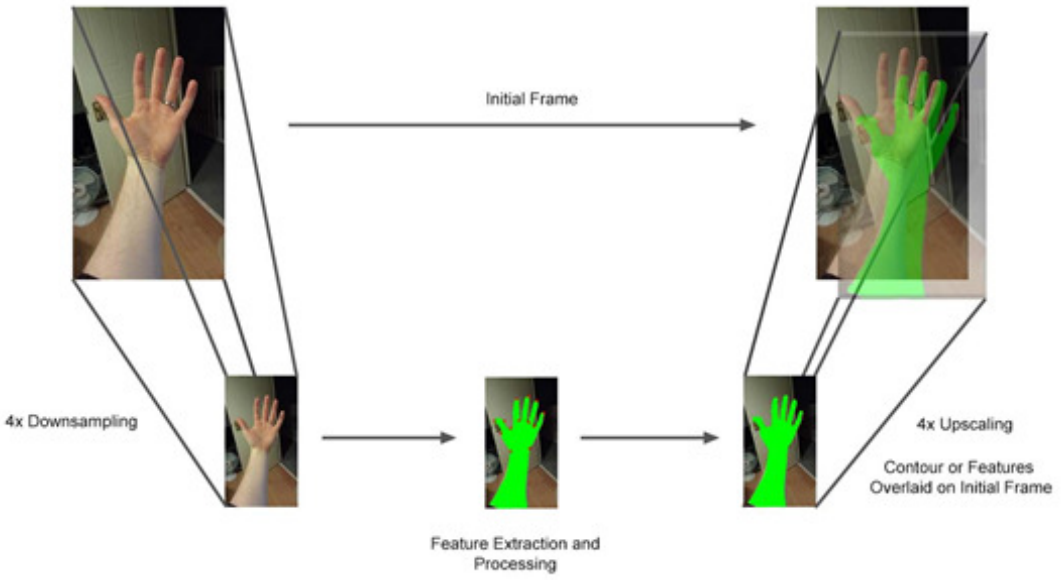


Fig. 2. Overview of our sign detection process.

camera and translating the detected hand pose into a letter using the machine learning technique called Support Vector Machine (SVM).

This intelligent vision system uses a regular optic camera such as a smartphone camera working with RGB flat image data, not requiring any type of calibration or invasive sensors.

On the other hand, our system processes images at an average rate of $170ms$ per image, addressing successfully the real-time constraint. Indeed, the visual signing rate is of 2.3-2.5 signs per second [14], enforcing real-time SLR systems to process each sign at $2fps$ or in less than $400ms$ per image.

The study developed in this work aims to enrich the lives of a wide range of people with hearing and/or speech impairments and their respective circles.

The strict use of free and open source technologies in this project along with the use of cheap, portable hardware such as a smartphone implies that the resultant prototype could be highly accessible to a large number of users.

The original contribution of our work is the study of the automated BSL recognition process and includes the creation of a BSL large-scale dataset of c. 13k samples as well as the design, development, and deployment of a new intelligent vision system for automated BSL recognition in real-world environment.

The paper is structured as follows. In Section II, we present our detection and recognition system for BSL, while in Section III we report and discuss the carried out experiments which results show the developed SLR system has excellent performance on real-world large-scale datasets, both in terms of accuracy and computational efficiency. Conclusions are drawn up in Section IV.

II. Our Method

The developed intelligent vision system embeds a two-step computational process. The first step involves computer-vision algorithms processing the input image (see Fig. 2) and results in the visual sign detection, as described in Section II-A. The second step consists in machine-learning algorithms using Support Vector Machine (SVM) to recognize the detected visual sign, as explained in Section II-B.

A. Sign Detection

Let us consider a colour RGB image or video frame $I(x, y)$ where M x N is its size, with M, its width, and N, its height, recorded live with the smartphone using the *OpenCV Camera Listener* triggered by our application.

In the first phase, the intelligent vision system running on the smartphone applies to the image $I(x, y)$ a series of mathematical operations as follows.

Firstly, the RGB image is transformed to the HSV colour space [15]. Secondly, the image is resized and down-sampled based on the Gaussian Pyramid as depicted in Fig. 2. Then, the image is segmented by thresholding [16], and the mask of the hand(s) is extracted by applying mathematical morphologic operations such as eroding and dilation [17]. Next, the Histogram of Oriented Gradients (HOG) [18] is computed as shown in Fig. 3. HOG assembles a histogram of prevailing, aggregated gradients throughout predefined blocks of an image. Because HOG produces such histogram, the number of features per vector is the same every time, given the input image $I(x, y)$ of a static size.

It is worth noting that through multiple pre-processing and cropping layers, a resultant image of a predictable size, i.e. M x N, is given to the HOG detector feature extraction layer.

Post-processing of the image encompasses any upscaling and resizing that may need to be done to render the correct

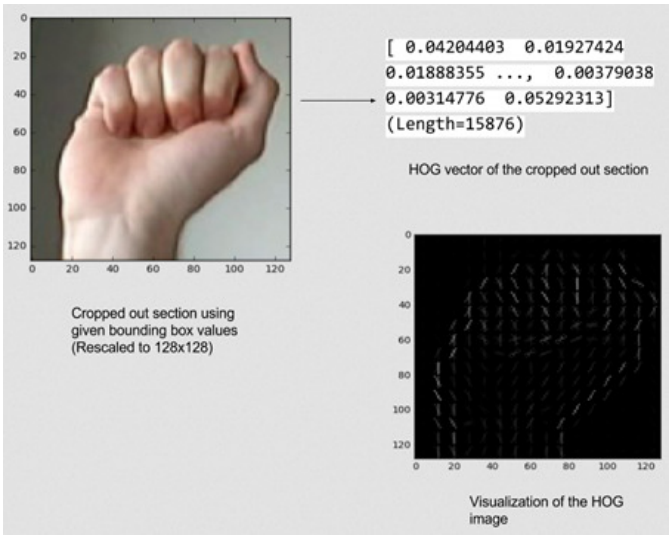


Fig. 3. Histogram of Oriented Gradients (HOG) visualisation.

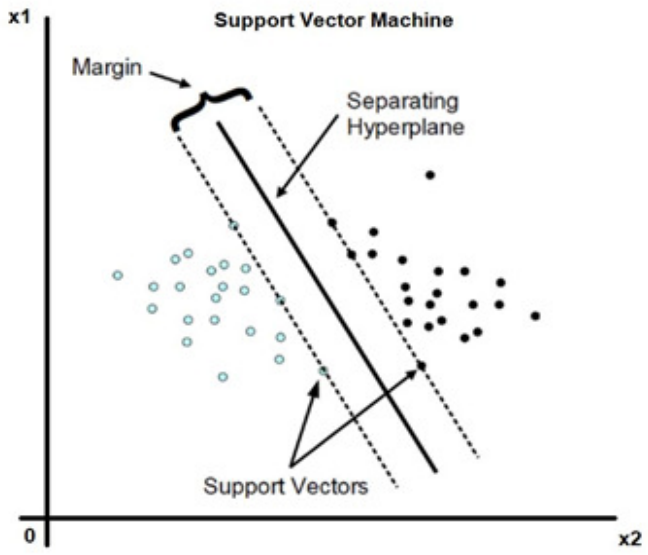
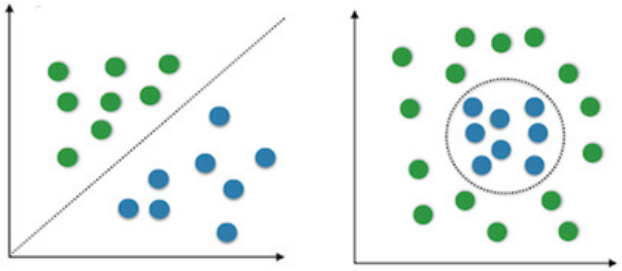


Fig. 4. Support Vector Machine (SVM) hyperplane visualisation.



(a) (b)

Fig. 5. SVM class separation using: (a) a linear kernel; (b) a radial basis function (RBF) kernel.

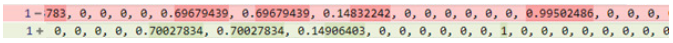


Fig. 6. Min-max normalisation performed on feature data before being fed into the SVM layer. It shows new maximum value of 1 on the lower row.

information. Hence, if contour overlays are required to indicate the detected hand shape (as displayed in Fig. 8), then upscaling the previously down-sampled materials is done in this stage of the process.

B. Sign Recognition

Once data has been processed into a set of frames containing only pertinent gesture data as described in Section II-A, these are analysed and fed into a model as explained in this Section II-B, in order to be classified in one of the 26 classes of the BSL alphabet.

In this work, the adopted classifier is a Support Vector Machine (SVM), since SVM is an efficient implementation of a supervised machine learning approach for classification and decision making [19], while requiring only a small sample of training data [20]. SVM method is described in the subsections, as follows.

1) SVM Hyperplane: At its core, the SVM method attempts to find an ideal, separating hyperplane $H0$ (such as schematized in Fig. 4) to divide a dataset of n points into separate classes y_i (e.g. class $y_i = +1$ and class $y_i = -1$), as follows:

$$H0 \equiv \mathbf{w}^T \mathbf{x_i} + b = 0, \tag{1}$$

with $\mathbf{x_i} = (x_1, x_2, ..., x_n)$, the input vector, $\mathbf{w} = (a, -1)$, the weight factor, b, the bias, and a such as $ax_1x_2 + b = 0$; this equation being derived from two-dimensional vectors, but in fact, works for any number p of dimensions.

The Support Vectors themselves are the data points closest to this plane of division and are therefore critical to segregating classes. Depending on the class (i.e. class +1 or class -1) they are part of, they belong either to $H1$ or $H2$, defined as:

$$H1 \equiv \mathbf{w}^T \mathbf{x_i} + b = 1, \tag{2}$$

$$H2 \equiv \mathbf{w}^T \mathbf{x_i} + b = -1, \tag{3}$$

with the (hard) margin defined as $D = H1 - H2$ (see Fig. 4); the hyperplane $H0$ being the median in between $H1$ and $H2$.

Consequently, each feature vector $\mathbf{x_i}$ is classified as follows:

$$\text{class } y_i = +1, \text{ if } \mathbf{w}\mathbf{x_i} + b \geq 1, \tag{4}$$

$$\text{class } y_i = -1, \text{ if } \mathbf{w}\mathbf{x_i} + b \leq -1. \tag{5}$$

For SLR, this means finding a hyperplane (Eq. 1) between data from an actual gesture (i.e. class $y_i = +1$) and interstitial hand movements which are not categorised (i.e. class $y_i = -1$), leading to a *one-versus-all* approach, and then using the hyperplane to make predictions as per Eqs. 4-5.

2) SVM Kernels and Hyperparameters: The SVM is an integral part of the application, and different possible kernels could be implemented in order to gain linear separation in the data space, in case data is linearly separable (linear SVM), or otherwise (non-linear SVM) in a higher dimensional space. In particular, Linear and Radial Basis Function (RBF) kernels were used to test the application as reported

in Section III. As implied by the name, the linear kernel ($K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)$) attempt to draw a line between class data (see Fig. 5 (a)), whereas the radial kernel ($K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$) tries to fit a curved or radial shape between classes (see Fig. 5 (b)). These are very different kernels, but can often produce similar results in practice.

In the finalised prototype, the RBF kernel has been adopted based on the resultant test data accuracy (see Table 1). After being trained offline with 20 samples per class, the SVM is run live within the smartphone application. The maximum iterations before terminating the SVM is set to 100. The parameters have been chosen by cross-validation on the training set.

3) Normalisation: Min-max normalisation has been performed on feature data before being fed into the SVM layer. Normalisation is important, since it flattens the data to an appropriate scale, in this case from 0 to 1.

The formula given for min-max normalisation is that a data point has the set's minimum subtracted from it, and then this value is divided by the set's maximum minus the set's minimum. Accordingly, a value A for B is then derived as:

$$A = \frac{B - \min(B)}{\max(B) - \min(B)}. \tag{6}$$

Figure 6 shows a small example of a row of matrix data being set to new normalised values by applying Eq. 6. Within this example, the far right non-zero, previously maximal value is set to the new scale maximal value which is 1. The other values in the matrix are then scaled according to their relation to this new maximal value.

III. Experiments and Discussion

Our intelligent vision system was validated by running a series of experiments and assessing them quantitatively as reported below.

The effectiveness of the prototype has been measured using the following metrics [5]:

$$\textit{precision } (P) = \frac{TP}{TP + FP}, \tag{7}$$

$$\textit{recall } (R) = \frac{TP}{TP + FN}, \tag{8}$$

$$\textit{specificity } (S) = \frac{TN}{TN + FP}, \tag{9}$$

$$\textit{accuracy } (ACC) = \frac{TP + TN}{TP + TN + FP + FN}, \tag{10}$$

where TP is the True positive rate, FP is the False Positive rate, FN is the False Negative rate, and TN is the True Negative rate.

Another common metric is the F1-Measure or F1 score which is the harmonic mean of the precision and recall and which could be used when a balance between precision and recall is needed and when the class distribution is uneven (i.e. high $TN + FP$). F1 score is defined as follows:

$$\textit{F1-Measure} = 2\frac{P * R}{P + R}. \tag{11}$$

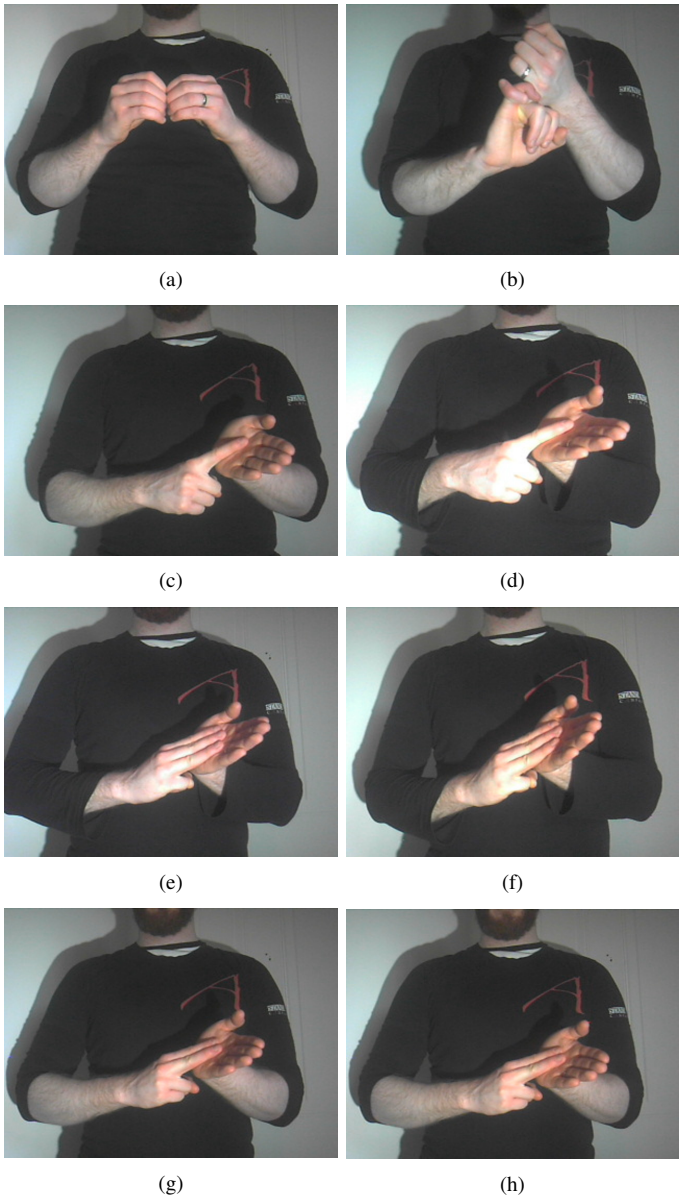
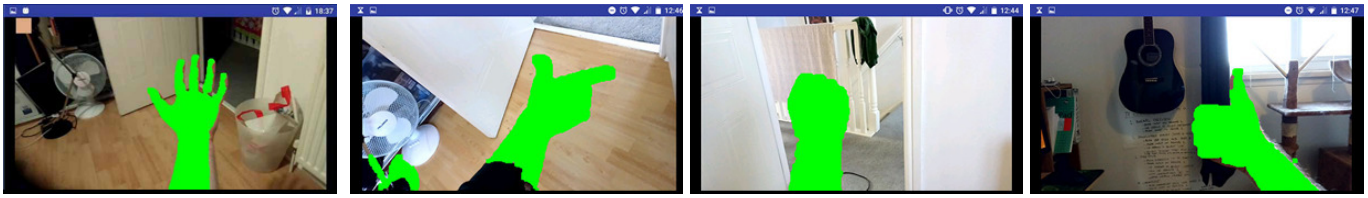


Fig. 7. Samples of our system performing successfully BSL sign recognition of the (a) B letter; (b) S letter; (c)-(d) L letter, in images with different illumination conditions; (e)-(f) M letter in images with different yaw rotations; (g)-(h) N letter in images with different pitch rotations.

As BSL fingerspelling datasets are difficult to find openly, a BSL dataset was created using the dataset creator application developed for this project.

This dataset is unique in the study, as it comprehends BSL phonemes consisting of two-handed gestures. Dataset samples are shown in Fig. 7. This BSL dataset contains 2,600 images of sign classes from A to Z. It is worth noting that H and J letter images were taken with last endpoint of motion gesture as an approximation. The images are in the Portable Network Graphics (.png) format and have a 640x480 size, a resolution of 72dpi, and a bit depth of 24. Moreover, the lighting is varying, and it also contains an even 50% mixture of signs



(a) (b) (c) (d)

Fig. 8. Samples of our system performing hand detection in images with (a) cluttered background; (b) patterned background; (c) high luminance; or (d) high exposure.

with sleeves up and down, in order to introduce variation to the training model.

The dataset has been created in Python language, using the *Dataset Creator* application. The Python language has been chosen due to its lightweight nature and portability, but also because OpenCV wrappers are available for it. Indeed, the OpenCV 'Cv2' library for Python is used to access the webcam data when creating the image dataset and is therefore well suited to the integration with the rest of the project applications.

The prototype is written in Java and uses OpenCV 3.4.3. as well as the Java Native Interface (JNI) platform to enable Java running on the Java Virtual Machine (JVM) to interact with native platform applications written in lower-level languages such as C++. Hence, this JNI interface for the OpenCV dependency provides a critical interface into low-level hardware operations to run OpenCV processes through the C++ library. This interface enables thus the more computationally expensive parts of the application to run more efficiently and thence, provides a better user experience on the low-end phone hardware.

The training of the classifier has been performed using 520 samples (i.e. 26 classes with 20 samples per class) on a computer with features such as AMD FX-8320 3.5GHz (8 cores, 8 threads), 32nm architecture, 8Gb dual-channel DDR3 @ 802MHz, Windows 10 Home, 931Gb Western Digital SATA (7.2k rpm).

For the training function of the Offline Trainer, the datasets used the same imaging kernel as our smartphone application and they were processed image by image using a recursive Image Runner class; this Image Runner taking an input directory and processing each image in the parent folder and any subfolder.

The testing of the prototype has been run on a ZTE Blade V7 smartphone. This is a low-end budget Android phone (sub-£100) for the purposes of encouraging efficient programming and aiding in the overall availability of the end product. This phone model's specifications are as follows: Android 6.0 (Marshmallow) OS, Chipset Mediatek MT6753 (28nm), Octa-core 1.3 GHz Cortex-A53 CPU, Mali-T720MP3 GPU, 16 GB, 2 GB RAM of internal memory, and camera features such as 13 MP, PDAF, Dual-LED dual-tone flash, HDR, panorama, with 1080p @30fps video recorder.

The testing function of the offline trainer is entirely automated. After a training run is complete, the associated test data is run through the previously trained model. These test images have been subtracted from the initial training dataset in order to not render the testing redundant. Each test image is classified with a predicted letter, and then evaluated against the actual class of the input image. With this data, the outcome of the tests in terms of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) rates are computed. The output of these test operations goes to result log files for analysis and computation of Eqs. 7-11 for each run. Average recognition results can be found in Tables 1-2.

Tests have been carried out in the wild and included 13,066 sample images. BSL uses mainly two hands to represent a letter of the alphabet. However, the letter 'C' is signed using one hand only. Thence, we performed tests for both one and two hands, as illustrated in Fig. 8.

We reported in Tables 1-2 the primary data obtained by processing the approach presented in [21] and our method, respectively, on the BSL large-scale dataset. The approach of [21] involves visual features such as Edge Orientation Histogram features (EOH), whereas our system uses Histogram of Oriented Gradients (HOG) features; the classifier being the Support Vector Machine (SVM), with a linear and a radial basis function (RBF) kernel, respectively.

We can observe that our method combining HOG features and the SVM classifier with a RBF kernel outperforms the state-of-the art approaches in regards of both recognition accuracy and computational efficiency, while performing in the wild.

Furthermore, our method has been compared to available secondary data in the literature. The work of [11] studies Genetic Algorithms (GA) as classifiers for gesture recognition (using only 6 different gesture classes), achieving 98.6% accuracy, but reaching time frameworks in the range of dozens of seconds for the overall image processing. Moreover, [11] requires hundreds of samples for training and has only been tested on 100 samples.

On the other hand, [8] uses a Hidden Markov Model (HMM) and has a BSL recognition accuracy rate per letter of 84.1%, whereas our BSL recognition accuracy rate per letter is 99% and our processing time of 170ms (i.e. less than the 400ms mentioned in the study of [14]) ensures a real-time visual sign recognition pace.

TABLE I
SIGN RECOGNITION PERFORMANCE USING DIFFERENT METHODS.

Method	Features	SVM Kernel	Precision	Recall	Specificity	Accuracy	F1-Measure
[21]	EOH	Linear	0.853	0.867	0.994	0.988	0.852
[21]	EOH	RBF	0.855	0.868	0.994	0.988	0.854
Our	HOG	Linear	0.863	0.874	0.994	0.989	0.861
Our	**HOG**	**RBF**	**0.869**	**0.880**	**0.995**	**0.990**	**0.867**

TABLE II
AVERAGE PROCESSING TIME OF THE DIFFERENT METHODS PERFORMING SIGN RECOGNITION.

Method	Features	SVM Kernel	Average Processing Time (s)
[21]	EOH	Linear	0.178
[21]	EOH	RBF	0.178
Our	HOG	Linear	0.173
Our	**HOG**	**RBF**	**0.170**

IV. CONCLUSIONS

The paper proposes an assistive technology performing British Sign Language (BSL) alphabet translation in real-time and in real-world conditions, with an accuracy of over 99%. The design aims to provide an inclusive and accessible solution consisting in an intelligent vision system for automated BSL fingerspelling recognition, without being invasive or financially expensive. The algorithms developed within this system include Histogram of Oriented Gradients (HOG) method and the Support Vector Machine (SVM) technique. The resulting smartphone application has been successfully tested on a large-scale dataset in the wild. Its excellent performance leads, on one hand, to an accessible, assistive HCI product for non-deaf people wishing to learn BSL and/or to communicate using BSL with deaf persons, and on the other hand, to a potential HRI product for companion robots having the task to assist hearing and/or speech impaired people.

REFERENCES

[1] Actiononhearingloss.org.uk, "Facts and Figures,," 2019, Available online at: https://www.actiononhearingloss.org.uk.

[2] BDA, "British Deaf Association,," 2019, Available online at: https://bda.org.uk.

[3] NRCPD, "The National Registers of Communication Professionals working with Deaf and Deafblind People,," 2019, Available online at: https://www.nrcpd.org.uk.

[4] D. Waters, R. Campbell, C.M. Capek, B. Woll, A.S. David, P.K. McGuire, M.J. Brammer, and M. MacSweeney, "Fingerspelling, signed language, text and picture processing in deaf-native signers: The role of the mid-fusiform gyrus," *NeuroImage*, vol. 35, no. 3, pp. 1287–1302, 2007.

[5] J.I. Olszewska, "Designing transparent and autonomous intelligent vision systems," in *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, 2019, pp. 850–856.

[6] G. Plouffe and A.-M. Cretu, "Static and dynamic hand gesture recognition in depth data using dynamic time warping," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 2, pp. 305–316, 2016.

[7] M. Goyal, B. Shahi, K.V. Prema, and N.V.S.S. Reddy, "Performance analysis of human gesture recognition techniques," in *Proceedings of the IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology*, 2017, pp. 111–115.

[8] S. Liwicki and M. Everingham, "Automatic recognition of fingerspelled words in British sign language," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 50–57.

[9] J.L. Raheja, A. Mishra, and A. Chaudhary, "Indian sign language recognition using SVM," *Pattern Recognition and Image Analysis*, vol. 26, no. 2, pp. 434–441, 2016.

[10] A.B. Jani, N.A. Kotak, and A.K. Roy, "Sensor based hand gesture recognition system for English alphabets used in sign language of deaf-mute people," in *IEEE SENSORS Proceedings*, 2018, pp. 1–4.

[11] D.-J. Li an Y.-Y. Li, J.-X. Li, and Y. Fu, "Gesture recognition based on BP neural network improved by chaotic genetic algorithm," *International Journal of Automation and Computing*, vol. 15, no. 3, pp. 267–276, 2018.

[12] S. Salian, I. Dokare, D. Serai, A. Suresh, and P. Ganorkar, "Proposed system for sign language recognition," in *Proceedings of the IEEE Conference on Computation of Power, Energy Information and Communication*, 2017, pp. 58–62.

[13] B.L. Loeding, S. Sarkar, A. Parashar, and A.I. Karshmer, "Progress in automated computer recognition of sign language," in *Proceedings of the International Conference on Computers for Handicapped Persons*. 2004, pp. 1079–1087, LNCS, Springer.

[14] E. Klima and U. Bellugi, *The Signs of Language*, Harvard University Press, 1979.

[15] M. Loesdau, S. Chabrier, and A. Gabillon, "Hue and saturation in the RGB color space," in *Proceedings of the International Conference on Image and Signal Processing*. 2014, vol. 8509, pp. 203–212, LNCS, Springer.

[16] C. Rouge, S. Shaikh, and J.I. Olszewska, "HD: Efficient Hand Detection and Tracking," in *Proceedings of the IEEE Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2016, pp. 291–297.

[17] J.I. Olszewska, C. De Vleeschouwer, and B. Macq, "Multi-feature vector flow for active contour tracking," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 721–724.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.

[19] C. Cortes and V.N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] P. Matykiewicz and J. Pestian, "Effect of small sample size on text categorization with support vector machines," in *Proceedings of the ACL Workshop on Biomedical Natural Language Processing (BioNLP)*, 2012, pp. 193–201.

[21] S. Nagarajan and T.S. Subashini, "Static hand gesture recognition for sign language alphabets using edge oriented histogram and multi-class SVM," *International Journal of Computer Applications*, vol. 82, no. 4, pp. 28–35, 2013.